Software Reliability Measurements through Combination Models: Approaches, Results, and a CASE Tool

Michael R. Lyu
ECE Department
The University of Iowa
Iowa City, IA 52242

Allen Nikora
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Abstract

We propose a general combination approach to address the predictive accuracy problem for software reliability modeling. Instead of relying on any single model, we apply several linear combinations of existing software reliability models, called component models, to form a series of linear-combination models. In an experimental investigation with industrial project data, this set of linear combination models was judged to perform better than the component models. For the purpose of automating the procedures in applying existing software reliability models and the linear-combination models, we further suggest a computer-aided software engineering tool, called CASRE, to measure software reliability systematically. This paper discusses the ideas, approaches, and experimental results of the combination models, as well as the high-level design, structure, and functionality of the CASRE tool.

1. Introduction and Background

The complexity and size of software systems are growing dramatically. This trend makes the measurement of software reliability one of the major challenges for software engineers. Such a measurement presents important indicators for the quality of software products, and provides insight into the software design process so that areas for improvement may be identified. Software reliability measurement involves a set of techniques that apply probability theory and statistical analysis to assess the achieved reliability of work products, both quantitatively and objectively. A software reliability model specifies the general form of the dependence of the failure process on the principal factors that affect it: fault introduction, fault manifestation, failure detection and recovery, fault removal, and operational environment[1].

Since the publication of the first software reliability model almost 20 years ago[2], over 40 of them are now known to exist in the literature[3], [1]. It is likely that many more unpublished models are in use. The primary focus of these models is to assess current reliability and forecast future reliability, based on rational assumptions for the application of statistical inference techniques to the observed failure data. The main difficulty in software reliability engineering practice is to analyze the particular context in which reliability measurement is to take place so as to decide *a priori* which model is likely to be trustworthy. It has been shown that there is no best software reliability model for every case under all circumstances [4], [5]. Practitioners are left in a dilemma as to which software reliability models to choose, which procedures to apply, and which prediction results to trust, while contending with varying software development practices.

2. Forming Linear Combination Models

Although the software reliability measurement problem, by its nature, involves significant uncertainties, we are motivated to propose a scheme that makes more accurate software reliability predictions, at least in an average sense, than the traditional approaches. As a result, we have formalized a general combination modeling approach that could be applied for more accurate software reliability measurements as follows:

1. Identify a basic set of models (called *component* models) with generally good performance.
2. Select models that tend to cancel out in their biased (if any) predictions.
3. Keep track of the software failure data with all the component models.
4. Apply certain criteria to weigh the selected component models and form one or several *linear combination models* for final predictions. The weights could be either constants or variables which are dynamically changed with time.

0730/3157/91/0000/0577/$01.00 © 1991 IEEE

The basic component models are selected from the traditional software reliability models. As an example to illustrate this combination approach, we chose Goel-Okumoto Model(GO)[6], Musa-Okumoto Model(MO) [7], and Littlewood-Verrall Model(LV) [8] as the three component models to form a set of linear combination models. Reasons for choosing these three component models are:

1. Their predictive validity has been observed in[5] and they have been widely used.
2. They represent different categories of models: GO represents the exponential shape non-homogeneous Poisson process (NHPP), MO represents the logarithmic shape NHPP model, and LV represents the inverse-polynomial shape Bayesian model.
3. Their predictive biases tend to cancel: GO tends to be optimistic, LV tends to be pessimistic, and MO might go either way.

As a result, we formulated a set of four combination models as follows:

1. *Equally-Weighted Linear Combination (ELC) Model*
   This model is formed by assigning the three component models a constant, equal weight[9]. Namely, ELC = $\frac{1}{3}GO + \frac{1}{3}MO + \frac{1}{3}LV$. These weightings remain constant and unchanged throughout the measurement process.

2. *Median-Oriented Linear Combination (MLC) Model*
   Instead of choosing the arithmetic mean for the prediction in ELC, the median value is used. In other words, each time when a prediction is called for, the component model whose predicted value is the median is selected as the output of this model.

3. *Unequally-Weighted Linear Combination (ULC) Model*
   The weightings are determined similar to *Program Evaluation and Review Technique* (PERT), i.e., the formulation of this model is $\frac{1}{6}O + \frac{4}{6}M + \frac{1}{6}P$, where O represents an optimistic prediction, P represents a pessimistic prediction, and M represents the median prediction.

4. *Dynamically-Weighted Linear Combination (DLC) Model*
   In this model, we use a *meta-predictor* to form a linear combination of several predictions with the weightings chosen in some optimal way (e.g., posterior probabilities)[4]. A Bayesian interpretation of "prequential likelihood" as *a posteriori* could be dynamically calculated in a long run or in a short time window to determine the weight assignments. For this model, the weighting function for each of the component models varies with time.

3. Project Data Applications and Results

Project data recently taken from the Jet Propulsion Laboratory (JPL) was investigated for the purpose of validating the proposed combination modeling approach[9]. These projects include: The Voyager project, the Galileo project, the Galileo Command and Data Subsystem (CDS), the Magellan project, and the Alaska SAR project. We have evaluated each model's performance with respect to some measurement criteria and ranked them across the ensemble of the failure data sets.

3.1 Model Comparison Criteria and Procedures

In order to compare different models objectively and quantitatively, four formally defined measures have been adopted[4], [10]. These measures - Accuracy, Bias, Trend, and Noise - represent various quantities for the quality of software reliability measurement from a particular model.

We also evaluated 15 possible component models through six selection criteria, which include[11]: 1) Model validity; 2) Ease of measuring parameters; 3) Quality of assumptions; 4) Applicability; 5) Simplicity; and 6) Insensitivity to noise. Six models were judged to perform well under these criteria[5]. These models are: Jelinski-Moranda Model (JM)[2], Goel-Okumoto Model(GO)[6], Musa-Okumoto Model (MO)[7], Duane Model (DU)[12], Littlewood Model (LM)[13], and Littlewood-Verrall Model (LV)[8]. We used these six models as the reference points against which the results obtained from the four proposed combinatorial models were compared. Each of these ten models was applied to the five JPL data sets and evaluated with respect to the four criteria mentioned in the preceding paragraph.

3.2 Model Evaluation Results for Each Project

Tables 1 – 5 present results of the four measures against the ten models for the five investigated projects. In these tables, numbers in each column represent the computed measure under each criterion, with ranks in parentheses corresponding to the models in rows. The last column, "rank", was determined by equally treating all the four criteria.

Voyager Flight Software (133 data points/starting data–2)					
model	accuracy	bias	trend	noise	rank
JM	-894.7(10)	.2994(9)	.0995(9)	∞ (9)	(10)
GO	-573.7(7)	.2849(6)	.0965(7)	13.81(4)	(7)
MO	-571.5(6)	.2849(6)	.0957(5)	9.225(3)	(6)
DU	-586.6(8)	.2703(5)	.2551(10)	8.402(1)	(7)
LM	-829.9(9)	.2994(9)	.0994(8)	∞ (9)	(9)
LV	-549.1(2)	.0793(1)	.0876(3)	24.51(8)	(2)
ELC	-554.0(3)	.2084(3)	.0872(2)	15.15(6)	(2)
ULC	-557.8(4)	.2438(4)	.0951(4)	12.64(5)	(4)
MLC	-570.3(5)	.2849(6)	.0962(6)	9.129(2)	(5)
DLC	-543.1(1)	.2078(2)	.0866(1)	17.47(7)	(1)

RECOMMENDED MODELS: 1. DLC 2. ELC 2. LV

Table 1: Voyager Data Application Results

Galileo Flight Software (224 data points/starting data–24)					
model	accuracy	bias	trend	noise	rank
JM	-1074(5)	.3378(6)	.4952(6)	2.607(4)	(5)
GO	-1075(7)	.3378(6)	.4954(7)	2.593(3)	(7)
MO	-1078(9)	.3379(9)	.5041(10)	2.395(1)	(10)
DU	-1098(10)	.1944(1)	.4618(5)	4.541(6)	(6)
LM	-1074(5)	.3382(10)	.4954(7)	2.624(5)	(9)
LV	-1051(4)	.2592(5)	.1082(1)	23.33(9)	(4)
ELC	-1019(2)	.1991(2)	.2781(3)	17.72(8)	(1)
ULC	-1035(3)	.2569(4)	.3271(4)	13.80(7)	(3)
MLC	-1077(8)	.3378(6)	.5017(9)	2.584(2)	(8)
DLC	-984.7(1)	.2159(3)	.2484(2)	23.96(10)	(2)

RECOMMENDED MODELS: 1. ELC 2. DLC 3. ULC

Table 2: Galileo Flight Data Application Results

Galileo CDS Flight Software (358 data points/starting data–152)					
model	accuracy	bias	trend	noise	rank
JM	-643.0(6)	.1783(6)	.3450(6)	4.042(8)	(8)
GO	-639.3(5)	.1783(6)	.3408(5)	3.908(7)	(6)
MO	-681.1(8)	.1700(2)	.4262(9)	2.673(4)	(6)
DU	-728.5(10)	.1748(5)	.4282(10)	2.287(1)	(8)
LM	-643.0(6)	.1784(8)	.3450(6)	4.042(8)	(10)
LV	-612.3(2)	.2581(10)	.2426(1)	2.564(2)	(1)
ELC	-618.7(3)	.1732(4)	.2855(3)	2.853(5)	(1)
ULC	-626.9(4)	.1599(1)	.3072(4)	2.958(6)	(1)
MLC	-681.1(8)	.1700(2)	.4261(8)	2.672(3)	(4)
DLC	-606.1(1)	.1845(9)	.2618(2)	11.19(10)	(5)

RECOMMENDED MODELS: 1. LV 1. ELC 1. ULC

Table 3: Galileo CDS Data Application Results

Magellan Flight Software (197 data points/starting data–50)					
model	accuracy	bias	trend	noise	rank
JM	-627.1(6)	.2968(6)	.2399(6)	1.007(1)	(5)
GO	-627.1(6)	.2968(6)	.2399(6)	1.007(1)	(5)
MO	-627.1(6)	.2969(8)	.2399(6)	1.007(1)	(8)
DU	-616.0(2)	.1858(1)	.2180(5)	2.003(6)	(1)
LM	-627.1(6)	.2969(8)	.2399(6)	1.009(5)	(9)
LV	-622.9(5)	.3483(10)	.1429(2)	5.563(10)	(10)
ELC	-619.1(3)	.2140(2)	.1400(1)	4.260(8)	(1)
ULC	-622.3(4)	.2461(4)	.1871(4)	3.363(7)	(5)
MLC	-627.1(6)	.2968(5)	.2399(6)	1.007(1)	(4)
DLC	-609.2(1)	.2396(3)	.1049(3)	4.982(9)	(3)

RECOMMENDED MODELS: 1. ELC 1. DU 3. DLC

Table 4: Magellan Data Application Results

Alaska SAR Ground Software (367 data points/starting data–67)					
model	accuracy	bias	trend	noise	rank
JM	-915.7(2)	.3023(1)	.0606(4)	1.587(4)	(1)
GO	-915.8(6)	.3023(1)	.0615(6)	1.540(3)	(5)
MO	-915.7(2)	.3023(1)	.0620(7)	1.395(1)	(1)
DU	-925.5(10)	.4249(10)	.0918(9)	1.650(6)	(9)
LM	-915.7(2)	.3023(1)	.0606(4)	1.589(5)	(3)
LV	-920.5(9)	.3672(9)	.1009(10)	3.189(10)	(10)
ELC	-916.2(8)	.3434(7)	.0586(2)	2.220(9)	(8)
ULC	-915.9(7)	.3209(6)	.0528(1)	1.853(7)	(7)
MLC	-915.7(2)	.3023(1)	.0620(7)	1.413(2)	(3)
DLC	-914.8(1)	.3468(8)	.0569(3)	1.949(8)	(6)

RECOMMENDED MODELS: 1. JM 1. MO

Table 5: Alaska SAR Data Application Results

3.3 Overall Assessment

Tables 6 and 7 list the performance comparisons for all the above five data sets. The overall comparison is done by using all four measures in Table 6, or by using the prequential likelihood measure (the "Accuracy" criterion) alone in Table 7, since it was judged to be the most important one. In general, we consider a model as being satisfactory if and only if it is ranked 4 or better out of the 10 models for a particular project. To extend this idea, we define a "handicap" value, which is calculated by subtracting 4 (the "par" value) from the rank of a model for each data set before its ranks being summed up in the overall evaluation. (Or subtract 20 from the "Sum of Rank" row in Tables 6 and 7.) A non-positive handicap value represents satisfactory overall preformance for the five data sets.

Summary of Model Ranking for Each Data by All Four Criteria								
model	data sets					sum	handi-cap	total
	1	2	3	4	5			
JM	(10)	(5)	(8)	(5)	(1)	29	+9	(6)
GO	(7)	(7)	(6)	(5)	(5)	30	+10	(7)
MO	(6)	(10)	(6)	(8)	(1)	31	+11	(8)
DU	(7)	(6)	(8)	(1)	(9)	31	+11	(8)
LM	(9)	(9)	(10)	(9)	(3)	40	+20	(10)
LV	(2)	(4)	(1)	(10)	(10)	27	+7	(5)
ELC	(2)	(1)	(1)	(1)	(8)	13	-7	(1)
ULC	(4)	(3)	(1)	(5)	(7)	20	0	(3)
MLC	(5)	(8)	(4)	(4)	(3)	24	+4	(4)
DLC	(1)	(2)	(5)	(3)	(6)	17	-3	(2)

Table 6: Overall Assessment by All Criteria

Summary of Model Ranking for Each Data by the Accuracy Measure Alone								
model	data sets					sum	handi-cap	total
	1	2	3	4	5			
JM	(10)	(5)	(6)	(6)	(2)	29	+9	(6)
GO	(7)	(7)	(5)	(6)	(6)	31	+11	(8)
MO	(6)	(9)	(8)	(6)	(2)	31	+11	(8)
DU	(8)	(10)	(10)	(2)	(10)	+40	+20	(10)
LM	(9)	(5)	(6)	(6)	(2)	28	+8	(5)
LV	(2)	(4)	(2)	(5)	(9)	22	+2	(3)
ELC	(3)	(2)	(3)	(3)	(8)	19	-1	(2)
ULC	(4)	(3)	(4)	(4)	(7)	22	+2	(3)
MLC	(5)	(8)	(8)	(6)	(2)	29	+9	(6)
DLC	(1)	(1)	(1)	(1)	(1)	5	-15	(1)

Table 7: Overall Assessment by Accuracy

There are several important points that we can observe from these summary tables:

1. In general, the set of combination models perform better than the set of single models. The best four models, when considering all four measuring criteria (Table 6), are exactly the four linear combination models. When considering the Accuracy criterion alone (Table 7), the best three models, DLC, ELC, ULC, also belong to the combination model set.
2. The DLC, ELC, and ULC Models perform well across all data sets except the last one (Alaska SAR). By evaluating the handicap value, it is noted that these three models are the only three models considered satisfactory for Table 6 for a non-positive handicap value. In Table 7, only DLC and ELC are considered satisfactory. Moreover, these two models beat the other single models by a significant number of "strokes."
3. The DLC and ELC Models are rather consistent. Most other models seem to perform well for a few data sets but poorly for other data sets, and the fluctuation in performance is significant. By preserving good properties from the three famous models with equal weightings, the ELC model, as expected, achieve an overall good performance. As to the DLC model, it is amazing enough to see it consistently producing the best accuracy measure for every data set. This is not surprising, though, since the DLC model is allowed to dynamically change its weightings according to the outcome of the accuracy measure. This further suggests that, when other measure is decided to be important, we could use that measure as the weighting criteria in forming the DLC model to get the best result.
4. The poor performance of LV for the last data set (Alaska SAR) might be the reason why the combination models do not perform well for this data, since LV is one of the selected component models. All the combination models, however, perform better than LV.

There are at least four potential extensions to the combination scheme we have just described and evaluated. These potentials are:

1. We can try to apply models other than GO, MO, LV as component models. If some models are judged to perform well in a particular data set, they should be the candidates for the component models to form a combination model.
2. We can use more than three models as component models. It is postulated that the more component models we apply, the better the prediction we could expect.
3. We can also apply various weighting schemes for the combination, subject to project criteria and engineering judgements. In other words, users should be able to determine the way a combination model is formed.
4. Finally, the combination models themselves could be used as component models to form another combination model.

It is noted that by applying more complicated procedures for software reliability measurement, the simplicity attribute of an individual model might be lost. As a result, insight into the software reliability engineering process becomes harder to obtain. However, the main theme of this approach, as stated before, is the *accuracy*

of the measurements and predictions. After all, most software reliability models are statistical inference techniques applied to failure data coming out of a software "black-box." In that regard, the proposed combination schemes and their extensions do not degrade any properties assumed in current software reliability practices.

Nevertheless, given a failure data set, the complexity required in searching a good combination model and the resulting computation tasks could become overwhelming. Due to the tedious computation-intensive tasks that might be involved in the selection and application of the component models to form various combination models for investigation, a computer-aided approach is inevitable. For this purpose, we propose a CASE tool, called Computer-Aided Software Reliability Engineering (CASRE) system, for an automatic and systematic approach in measuring software reliability.

4. The CASRE Tool – Structure and Functionality

Figure 1 shows the proposed high-level architecture for CASRE, whose major functional areas are:

- Data Modification
- Failure Data Analysis
- Modeling and Measurement
- Modeling/Measurement Results Display

Much of CASRE's functionality is available in current software reliability tools[14], [15]. However, a feature unique to CASRE allows users to combine the results of several models in addition to executing a single model. Feedback from the Model Evaluation block assists users in identifying a model or combination of models best suited to the failure data being analyzed. Moreover, the i/o facility, the user interface, and the measurement procedures are greatly enhanced in this tool.

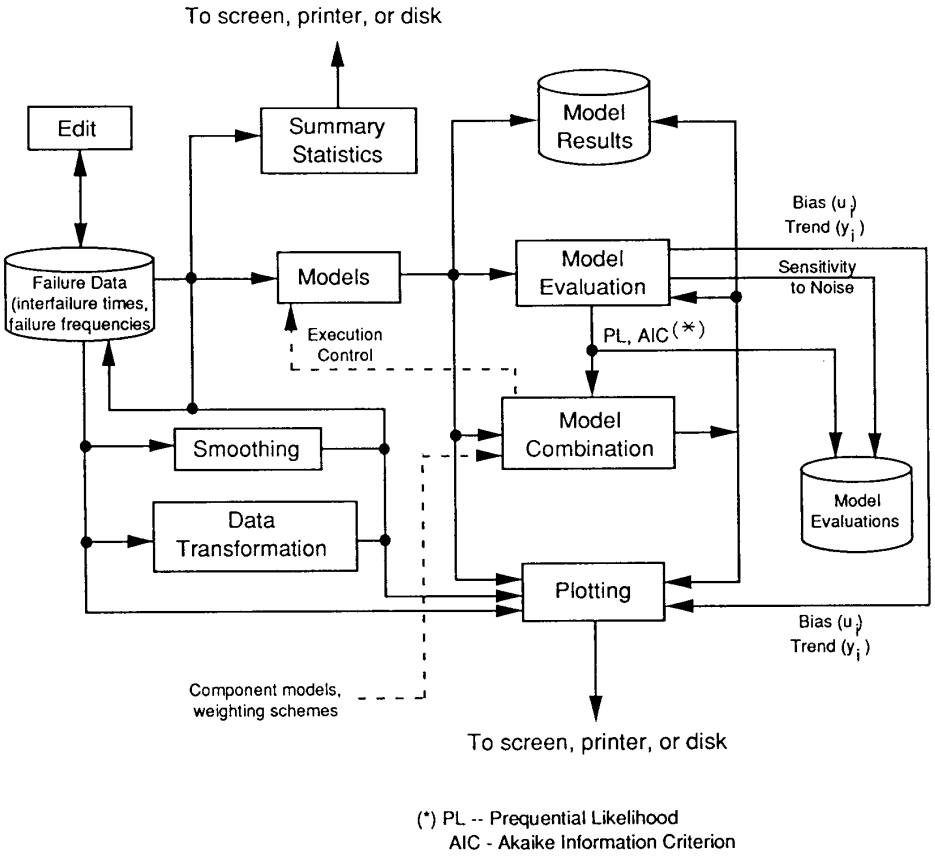

Figure 1: CASRE High-Level Architecture

4.1 Data Modification

CASRE allows users to create new failure data files, modify existing files, and perform global operations on files.

- Editing

CASRE allows users to create or alter failure history data files. A simplified spreadsheet-like user interface allows users to enter time between failures or test interval lengths and failure counts from the keyboard. Users are also allowed to invoke a preferred editor (e.g. emacs or vi).

- Smoothing

Since input data to the models is often fairly noisy, the following smoothing techniques are proposed:

- Sliding rectangular window
- Hann window
- Polynomial fit
- Specific cubic-polynomial fits (e.g. B-Spline, Bezier Curve)

Users select smoothing techniques appropriate to the failure data being analyzed. The smoothed input data can be plotted, used as input to a reliability model, or written out to a new file for later use. Summary statistics for the smoothed data can also be displayed (see "Failure Data Analysis" below).

- Data Transformation

In some situations, logarithmic, exponential, or linear transformations of the failure data produce better results. The following operations, currently available in some tools, allow users to transform an entire set of failure data in this manner.

- log(a * x(i)) + b); x(i) represents a failure data item, and a and b are user-selectable scale factors
- exp(a * x(i) + b)
- x(i) ** a
- x(i) + a
- x(i) * a
- user-specified transformation

As with smoothing, users select a specific transformation. Users are able to manipulate transformed data as they would smoothed data.

4.2 Failure Data Analysis

The "Summary Statistics" block in Figure 1 allows users to display the failure data's summary statistics, including the mean and median of the failure data, 25% and 75% hinge points, skewness, and kurtosis[16].

4.3 Modeling and Measurement

Figure 1 shows two modeling functions. The "Models" block executes single software reliability models on a set of failure data. The "Model Combination" block allows users to execute several models on the failure data and combine the results of those models. We include this capability because our experience in combining the results of more than one model indicates that such "combination models" may provide more accurate reliability predictions than single models. The block labeled "Model Evaluation" allows users to determine the applicability of a model to a set of failure data.

- Single Model Execution

Based on our experience in applying software reliability models, we include the following models in CASRE:

1) Bayesian Jelinski-Moranda Model (BJM) [18]
2) Brooks and Motley Model (BM)[19]
3) Duane Model (DU) [12], [20]
4) Geometric Model (GM) [19]
5) Goel-Okumoto (GO) [6]
6) Jelinski-Moranda (JM) [2], [21]
7) Keiller-Littlewood Model (KL)[22], [23]
8) Littlewood Model (LM) [13]
9) Littlewood non-homogeneous Poisson Process Model (LNHPP) [4]
10) Littlewood-Verrall (LV) [8]
11) Musa-Okumoto (MO) [7]
12) Generalized Poisson Model (PM)[19]
13) Schneidewind Model (SM)[24]
14) Yamada Delayed S-Shape Model (YM) [25]

The models should accept input in the form of interfailure times or failure frequencies.

CASRE allows users to choose the parameter estimation method (maximum likelihood, least squares, or method of moments). Model outputs include:

- Current estimates of failure rate/interfailure time
- Current estimates of reliability
- Model parameter values, including high and low parameter values for a user-selectable confidence bound

- Current values of the pdf and cdf
- The probability integral transform u_i[4]
- The normalized logarithmic transform of u_i, y_i[4]

Users can display these quantities on-screen or write them to disk.

• Combination Models

CASRE allows users to combine the results of several models according to the ELC, MLC, ULC, or DLC schemes. Users may also be allowed to define their own weighting schemes. The resulting combination models could be further used as the component models to form another combination model.

• Model Evaluation

CASRE includes the following statistical methods to help users determine the applicability of a model (including "combination models") to a specific failure data set:

- Computation of prequential likelihood (PL) function (the "Accuracy" criterion).
- Determination of the probability integral transform u_i, (plotted as the u-plot - the "Bias" criterion).
- Computation of y_i to produce the y-plot (the "Trend" criterion).
- Noisiness of model predictions (the "Noise" criterion).

The Akaike Information Criterion (AIC)[26], similar in concept to prequential likelihood, could also be implemented. This model evaluation function would also compute goodness-of-fit measures (e.g. Chi-Square test). The PL and AIC outputs are used as input to "Model Combination" to determine the relative contribution of individual models if the user has specified a combination model.

4.4 Display of Results

CASRE graphically displays model results in the following forms:

- Interfailure times/failure frequencies, actual and estimated
- Cumulative failures, actual and estimated
- Reliability growth, actual and estimated

Actual and estimated quantities are available on the same plot. Plots include user-specified confidence limits. Users are able to control the range of data to be plotted as well as the usual cosmetic aspects of the plot (e.g. X and Y scaling, titles). In a windowing environment, multiple plots could be simultaneously displayed. CASRE allows users to save plots displayed on-screen as a disk file or to print them. One public-domain tool, SMERFS [14] version 4, can write the data used to produce a plot to a file that can be imported by a spreadsheet, a DBMS, or a statistics package for further analysis. CASRE includes this capability.

The plotting function also produces u-plots and y-plots from Model Evaluation's u_i and y_i outputs. These plots indicate the degree and direction of model bias and the way in which the bias changes over time.

5. Conclusions and Future Work

We have proposed a set of linear-combination models for more accurate measurement of software reliability. These models have shown promising results when compared with the traditional single-model approaches. To relieve the tedious work involved in applying these approaches, a CASE tool (CASRE) is proposed to fully automate the software reliability measurement task. For the purpose of model validation and determining tool applicability, we need to obtain enough data to compare software reliability models and predictions across various types of software projects. In future investigations, we will apply more data sets to the proposed combination models for the purpose of validating them, and for refining the structure and functionality of the CASRE tool.

Acknowledgement

The research described in this paper was carried out at the University of Iowa under a faculty starting fund, and at the Jet Propulsion Laboratory, California Institute of Technology, through the Director's Discretionary Fund.

References

1. J. D. Musa, A. Iannino, and K. Okumoto, *Software Reliability – Measurement, Prediction, Application*, McGraw-Hill Book Company, New York, New York, 1987.
2. Z. Jelinski and P.B. Moranda, "Software Reliability Research," in *Statistical Computer Performance Evaluation*, ed. W. Freiberber, pp. 465-484, Academic, New York, 1972.
3. M.L. Shooman, "Software Reliability: A Historical Perspective," *IEEE Transactions on Reliability*, vol. R-33, no. 1, pp. 48-55, 1984.

4. A.A. Abdel-Ghaly, P.Y. Chan, and B. Littlewood, "Evaluation of Competing Software Reliability Predictions," *IEEE Transactions on Software Engineering*, vol. SE-12, pp. 950-967, September 1986.

5. M.R. Lyu, "Measuring Reliability of Embedded Software: An Empirical Study with JPL Project Data," in *Proceedings International Conference on Probabilistic Safety Assessment and Management*, Beverly Hills, California, February 1991.

6. A.L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," *IEEE Transactions on Reliability*, vol. R-28, pp. 206-211, 1979.

7. J.D. Musa and K. Okumoto, "A Logarithmic Poisson Execution Time Model for Software Reliability Measurement," in *Proceedings Seventh International Conference on Software Engineering*, pp. 230-238, Orlando, Florida, 1984.

8. B. Littlewood and J.L. Verrall, "A Bayesian Reliability Growth Model for Computer Software," *Journal Royal Statistics Society C*, vol. 22, pp. 332-346, 1973.

9. M.R. Lyu and A. Nikora, "A Heuristic Approach for Software Reliability Prediction: The Equally-Weighted Linear Combination Model," in *Proceedings 1991 International Symposium on Software Reliability Engineering*, Austin, Texas, May 1991.

10. A.P. Dawid, "Statistical Theory: The Prequential Approach," *Journal Royal Statistics Society A*, vol. 147, pp. 278-292, 1984.

11. A. Iannino, J.D. Musa, K. Okumoto, and B. Littlewood, "Criteria for Software Reliability Model Comparisons," *IEEE Transactions on Software Engineering*, vol. SE-10, no. 11, pp. 687-691, November 1984.

12. J.T. Duane, "Learning Curve Approach to Reliability Monitoring," *IEEE Transactions on Aerospace*, vol. AS-2, pp. 563-566, 1964.

13. B. Littlewood, "Stochastic Reliability Growth: A Model for Fault-Removal in Computer Programs and Hardware Designs," *IEEE Transactions on Reliability*, vol. R-30, pp. 313-320, October 1981.

14. W.H. Farr and O.D. Smith, "Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) User's Guide," TR 84-373, Revision 1, NSWC, December 1988.

15. B. Littlewood, A.A. Abdel-Ghaly, and P.Y. Chan, "Tools for the Analysis of the Accuracy of Software Reliability Predictions," in *Software System Design Methods*, pp. 299-335, Springer-Verlag, Heidelberg, 1986.

16. R.V. Hogg and A.T. Craig, *Introduction to Mathematical Statistics*, MacMillan, New York, 1978.

17. H. Joe and N. Reid, "Estimating the Number of Faults in a System," *Journal American Statistics Association*, vol. 80, pp. 222-226, March 1985.

18. B. Littlewood and A. Sofer, "A Bayesian Modificatin to the Jelinski-Moranda Software Reliability Model," *Software Engineering Journal*, vol. 2, pp. 30-41, 1987 .

19. W.H. Farr, "A Survey of Software Reliability Modeling and Estimation," Technical Report 82-171, NSWC, 1983.

20. L.H. Crow, "Confidence Interval Procedures for Reliability Growth Analysis," Technical Report 197, U.S. Army Material Systems Analyusis Activity, Aberdeen, Maryland, 1977.

21. M. Shooman, "Operational Testing and Software Reliability During Program Development," in *Proceedings 1973 IEEE Symposium on Computer Software Reliability*, pp. 51-57, New York, April 1973.

22. P.A. Keiller, B. Littlewood, D.R. Miller, and A. Sofer, "Comparison of Software Reliability Predictions," in *Proceedings 13th International Symposium on Fault-Tolerant Computing*, pp. 128-134, 1983.

23. P.A. Keiller, B. Littlewood, D.R. Miller, and A. Sofer, "On the Quality of Software Reliability Predictions," in *Proceedings NATO ASI Electronic Systems Effectiveness and Life Cycle Costing*, pp. 441-460, Berlin: Springer, Norwich, England, 1983.

24. N.F. Schneidewind, "Analysis of Error Processesin Computer Software," in *Proceedings International Conference on Reliable Software*, pp. 337-346, Los Angeles, 1975.

25. S. Yamada, M. Ohba, and S. Osaki, "S-Shaped Reliability Growth Modeling for Software Error Detection," *IEEE Transactions on Reliability*, vol. R-32, pp. 475-478, December 1983.

26. H. Akaike, "A New Look at Statistical Model Identification," *IEEE Transactions on Automatic Control*, vol. AC-19, pp. 716-723, 1974.